Exploiting Universum Data in AdaBoost Using Gradient Descent

Jingsong Xu[a], Qiang Wu[b], Jian Zhang[c,*], Zhenmin Tang[a]

[a] *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[b] *School of Computing and Communications, University of Technology, Sydney, NSW 2007, Australia*
[c] *Advanced Analytics Institute and School of Software, University of Technology, Sydney, NSW 2007, Australia*

Abstract

Recently, Universum data that does not belong to any class of the training data, has been applied for training better classifiers. In this paper, we address a novel boosting algorithm called $\mathfrak{U}$adaBoost that can improve the classification performance of AdaBoost with Universum data. $\mathfrak{U}$adaBoost chooses a function by minimizing the loss for labeled data and Universum data. The cost function is minimized by a greedy, stagewise, functional gradient procedure. Each training stage of $\mathfrak{U}$adaBoost is fast and efficient. The standard AdaBoost weights labeled samples during training iterations while $\mathfrak{U}$adaBoost gives an explicit weighting scheme for Universum samples as well. In addition, this paper describes the practical conditions for the effectiveness of Universum learning. These conditions are based on the analysis of the distribution of ensemble predictions over training samples. Experiments on handwritten digits classification and gender classification problems are presented. As exhibited by our experimental results, the proposed method can obtain superior performances over the standard AdaBoost by selecting proper Universum data.

Keywords: AdaBoost, gradient boost, $\mathfrak{U}$adaBoost, Universum, $\mathfrak{U}$-SVM

1. Introduction

Conventional machine learning algorithms take labeled data, unlabeled data or both of them for learning. Vapnik [1] proposed the third kind of data: Universum data. The Universum data contains the data that belongs to the same domain as the classification problem but it does not belong to any class of the problem. For example, in handwritten digits recognition problems, if the samples of handwritten digits '5' and '8' are prepared for learning, then other handwritten digit samples can be naturally treated as Universum data since they belong to the same domain but cannot be assigned to any of the two classes.

It is a common case that large labeled training data is included in order to obtain good quality of training. However, it is quite costly or sometime even impossible to have very large training data. To deal with such problem, semi-supervised learning is a common option when unlabeled data is available since unlabeled data helps model data distribution of the whole data. On the other hand, without unlabeled data, Universum data is still able to provide the supports to maintain the training quality with relatively small labeled data set. The reason is Universum data can be generated through a lot of ways from labeled data only [2] (mentioned later). Moreover, Universum data can carry additional valuable prior information from the domain of the problem into the training process. To the best of our knowledge, there is no comparison between semi-supervened learning and Universum based learning. But in our opinion, Universum based learning can better model the whole data set since Universum data stays in the same domain of learning problem with which we are concerned [1] while the unlabeled data may be too general and stay outside of the domain. In terms of data acquisition, Universum data can be obtained more widely.

Vapnik first discussed transductive learning with Universum since transductive learning provides prior information to estimate the upper bound of inductive inference [1]. However, the classifier trained by inductive learning is more practical to classify unknown data. Weston *et al.* [2] proposed an inductive algorithm, Universum Support Vector Machines ($\mathfrak{U}$-SVM). $\mathfrak{U}$-SVM contains an additional regularization term for Universum data in addition to conventional SVM. The regularization is based on this assumption: the decision values on the Universum data should be close to zero. That is Universum data should fall inside the margin of the classifier since it does not belong to any class. The Universum samples which meet such assumption are called contradictions because the goal of learning is putting labeled data outside of the margin. Thus the margin should contain more Universum data to achieve better learning performance. More Universum data means more contradictions. Therefore the learning criterion for Universum based learning is called Maximal Contradiction on Universum (MCU) [1]. Two learning problems: common semi-supervised and training

*Corresponding author
Email addresses: xjsxujingsong@gmail.com (Jingsong Xu), Qiang.Wu@uts.edu.au (Qiang Wu), Jian.Zhang@uts.edu.au (Jian Zhang), tzm.cs@mail.njust.edu.cn (Zhenmin Tang)

based on Universum are demonstrated in Fig. 1. The unlabeled data should be away from the margin like labeled data while Universum data should fall between margins.

Sinz *et al.* [3] analyzed the 𝔘-SVM for inference and they showed that 𝔘-SVM would give the hyperplane which had its normal lying in the orthogonal to the principal directions of Universum data. They also discussed the connection of least squared version of 𝔘-SVM with fisher discriminant analysis and oriented principal component analysis. They showed that 𝔘-SVM outperformed SVM with carefully selected Universum data. In addition to SVM, Universum data has also been extended to other learning problems, such as semi-supervised learning [4], linear discriminant analysis [5], twin support vector machine [6], cost-sensitive learning [7], linear programming [8], domain adaptation [9]. In terms of application, besides the handwritten digits recognition problem mentioned above, it is also applied into medical imaging [10], document clustering [11], pose recognition [12] [13], etc.

Universum data is always obtained from the domain of the classification problem mentioned above. Weston *et al.* [2] proposed four kinds of Universum data: random noise, the rest of the training data (e.g. the other digits in handwritten digits recognition problem), artificial data from the same distribution of training data and random average of training data. Bai & Cherkassky [14] applied Universum data into gender classification and they took three kinds of Universum data: random average, empirical distribution and animal faces. In the field of Universum data selection, Sinz *et al.* [3] suggested that a good Universum set should contain invariant directions and be positioned "in between" the two classes of the classification problem. Chen & Zhang [15] proposed a guided formulation to pick the informative ones, i.e., in-between Universum (IBU) samples.

Boosting family contains a series of well-known algorithms with a large number of applications. Motivated by the success of 𝔘-SVM, Shen *et al.* [16] proposed 𝔘Boost by adding universum data to boosting algorithms and showed that they can benefit from Universum data as SVM did. 𝔘Boost is derived from AdaBoost-CG [17] which is another view of boosting. Compared with AdaBoost-CG, AdaBoost is a stagewise method [18] which is more general and popular. The whole training procedure of AdaBoost is also much faster. Although Universum data has shown its power on 𝔘-SVM [2] and 𝔘Boost [16], to our knowledge, its importance on AdaBoost has not been evaluated.

In this paper, we propose a new boosting algorithm called 𝔘adaBoost to improve the classification performance of AdaBoost with the help of Universum data. The learning is not straight forward since Universum data belongs to neither positive nor negative data. Stagewise AdaBoost keeps the pre-selected weak classifiers unchanged in the following training. It pays more attention on misclassified samples in the next training iteration. The weights for training samples and coefficients for the pre-selected weak classifiers are obtained according to gradient descent. Involving Universum data into AdaBoost framework needs to take these properties of AdaBoost into account. Instead of 𝔘adaBoost, 𝔘Boost takes Universum as an conventional convex optimization problem and solves it by column generation as AdaBoost-CG does.

To tackle above challenges, we propose explicit weighting schemes for both labeled data and Universum data which are both involved in AdaBoost training procedure. The rationale of updating weights in common AdaBoost training is to enforce the training to focus on hard samples. In this paper, this rationale is revisited and further extended on Universum data in the proposed 𝔘adaBoost.

The major contributions of this paper are as follows:

1) Given Universum data, a new 𝔘adaBoost learning based on the framework of AdaBoost is proposed. We propose 𝔘adaBoost using the same functional gradient descent as AdaBoost. By taking advantage of AdaBoost, 𝔘adaBoost is much easier and more practical to be applied than 𝔘Boost [16] since 𝔘adaBoost only needs one parameter to tune.

2) The whole training procedure of 𝔘adaBoost is efficient. The time cost for 𝔘adaBoost is less than 𝔘Boost. Our experimental results demonstrate such improvement.

3) 𝔘adaBoost provides a better framework for investigating the benefits of Universum data in boosting approaches. It is known that AdaBoost is a popular algorithm in boosting algorithm family. In recent years, researches have contributed significant efforts to investigate AdaBoost in order to improve its performance. To our best knowledge, 𝔘adaBoost is the only framework using the same approach (i.e. stagewise) as AdaBoost and integrating Universum data, so the performance evaluation on integrating Universum data into AdaBoost is more precise and convincing. In contrast, 𝔘Boost follows column generation approach [17] which is different from AdaBoost so we cannot use 𝔘Boost framework to evaluating benefits of Universum data to AdaBoost.

4) Also, in this paper, we discuss a method for selecting effective and informative Universum data in order to better take the advantages of Universum data in AdaBoost framework. This will benefit several applications in computer vision area.

The paper is structured as follows. In section 2, we discuss the related work about 𝔘-SVM, 𝔘Boost and the motivation to 𝔘adaBoost. In section 3, we propose the novel boosting formulation 𝔘adaBoost based on the Universum data and compare it with semi-supervised boosting, AdaBoost and 𝔘Boost. In section 4, we analyze the practical conditions for 𝔘adaBoost. In section 5, the performance of our model will be demonstrated with several public data sets. In section 6, we conclude the paper.

2. Related Works and Motivation

Notations: In this paper, our focus is only on binary classification problems, while our method can be extended to the multi-class scenario. Let $\mathbf{X}_{\mathfrak{L}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$ be the set of m labeled examples, where $y_i \in \{-1, 1\}$ is the class label. Let $\mathbf{X}_{\mathfrak{U}} = \{\mathbf{x}_1^*, \mathbf{x}_2^*, ..., \mathbf{x}_n^*\}$ represent the Universum data with n samples. w_i and w_j^* represent the weights of the labeled sample $(\mathbf{x}_i, y_i)$ and $\mathbf{x}_j^*$ during boost training phase respectively.

2.1. $\mathfrak{U}$-SVM and $\mathfrak{U}$Boost

Weston *et al.* [2] proposed $\mathfrak{U}$-SVM and treated it as an inductive learning problem. The Universum examples are considered to be close to the separating hyperplane selected by SVM. The optimization objective should minimize the cumulative loss on the Universum examples. Given the Hinge loss $H_a[t] = \max\{0, a-t\}$ for the standard SVM ($a = 1$) and ε-insensitive loss $I_\varepsilon[t] = H_{-\varepsilon}[t] + H_{-\varepsilon}[-t]$ for Universum data, the learning problem can be formulated as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C_{\mathfrak{L}} \sum_{i=1}^{m} H_1[y_i f_{\mathbf{w},b}(\mathbf{x}_i)] + C_{\mathfrak{U}} \sum_{j=1}^{n} I_\varepsilon[f_{\mathbf{w},b}(\mathbf{x}_j^*)] \tag{1}$$

The first two terms are for the standard SVM and the last one is for Universum data. $C_{\mathfrak{L}}$ and $C_{\mathfrak{U}}$ are parameters for regularization. $f_{\mathbf{w},b}(\mathbf{x})$ is the learned classifier. Parameter ε controls the margin of Universum data. Sinz *et al.* [3] gave the least squares version of $\mathfrak{U}$-SVM and showed that this kind of $\mathfrak{U}$-SVM can also show similar performances with the original $\mathfrak{U}$-SVM with less parameters (there is no ε in Eq.(2)).

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C_{\mathfrak{L}}}{2} \sum_{i=1}^{m} Q_{y_i}[f_{\mathbf{w},b}(\mathbf{x}_i)] + \frac{C_{\mathfrak{U}}}{2} \sum_{j=1}^{n} Q_0[f_{\mathbf{w},b}(\mathbf{x}_j^*)] \tag{2}$$

where $Q_a[t] = \|t-a\|_2^2$ is the quadratic loss for labeled data ($a = y_i$) and Universum data ($a = 0$).

Shen *et al.* [16] proposed a boosting algorithm $\mathfrak{U}$Boost using Universum data. They applied squared loss of Universum data in the optimization objective. They formulated this problem with the same framework of AdaBoost-CG [17] by adding the regularization term for Universum data.

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \exp(z_i) + \frac{c}{2n} \sum_{j=1}^{n} z_j^{*2} + D\mathbf{1}^\top \mathbf{w} \\ &\text{s.t. } z_i = -y_i \mathbf{H}_i \mathbf{w}, \forall i; z_j^* = \mathbf{H}_j^* \mathbf{w}, \forall j; \mathbf{w} \succeq \mathbf{0}. \end{aligned} \tag{3}$$

D controls the regularization of the weighting coefficients of boosting algorithm. c controls the trade-off between the errors of labeled data and unlabeled data. $\mathbf{H}$ is the prediction of all available weak classifiers on the training data. $\mathbf{H} = \{h_k(\mathbf{x}) : \mathbf{x} \to \{1, -1\}\}$, $H_{ik} = h_k(\mathbf{x}_i)$. h_k denotes the k-th weak classifier. $\mathbf{H}_i$ is the i-th row of $\mathbf{H}$ which denotes the output of all weak classifiers on example $\mathbf{x}_i$. Likewise, $\mathbf{H}^*$ is the prediction of Universum data. Two additional variables z and z^* are used for generating the dual problem. Based on this formulation, they obtained its corresponding dual problem and solved it with column generation. Experiments on a variety of handwritten digits recognition problems and computer vision problems showed that $\mathfrak{U}$Boost outperformed AdaBoost and AdaBoost-CG.

2.2. Motivation

Although $\mathfrak{U}$Boost has shown the effectiveness of Universum data for boosting algorithms, it still has some drawbacks. First, the selection of the trade-off parameter D of Eq.(3) in AdaBoost-CG influences the final performance significantly and how to tune the parameter efficiently remains an unsolved problem [17]. There are two parameters D and c in $\mathfrak{U}$boost, resulting in more difficulties to select a pair of appropriate parameters. Second, $\mathfrak{U}$Boost needs to update all the previous trained weighting coefficients w during each iteration, making the training procedure more and more slowly. Last, $\mathfrak{U}$Boost is built on AdaBoost-CG, therefore, the direct performance comparison is between $\mathfrak{U}$Boost and AdaBoost-CG other than AdaBoost. It is difficult to verify the effectiveness of Universum learning with conventional boosting algorithm.

In this paper, we propose a new boosting algorithm with Universum data, termed $\mathfrak{U}$adaBoost, to overcome these limitations. First, $\mathfrak{U}$adaBoost is derived by margin cost function framework of boosting algorithms which gives explicit weighting schemes for labeled and Universum data. $\mathfrak{U}$adaBoost trains the coefficient of the selected weak classifier at current iteration while $\mathfrak{U}$Boost updates coefficients of both newly selected weak classifier and also the previously determined weight values of existing weak classifiers. Therefore, $\mathfrak{U}$adaBoost is much easier and faster to train than $\mathfrak{U}$Boost. In addition, only one parameter c for the loss of Universum data in $\mathfrak{U}$adaBoost makes it more practical to use. Finally, since $\mathfrak{U}$adaBoost follows the same stagewise way to learn as AdaBoost, it is easier to understand the difference of the AdaBoost with and without Universum data. This is very important since we can predict the performance of boosting algorithm with different kinds of Universum data before really training the final classifier.

Cherkassky *et al.* [19] proposed practical conditions to verify the effectiveness of Universum data. Histogram is applied as an analytical tool. They showed that the projection of effective Universum data on the norm direction of the standard SVM decision boundary should be symmetric and has a wide distribution between the margins. In addition, they analyzed these conditions and showed that they were closely related to analytic conditions in [3]. Since the proposed $\mathfrak{U}$adaBoost follows the marginal framework of boosting, we can extend these conditions on $\mathfrak{U}$-SVM to $\mathfrak{U}$adaBoost. Experiments show that the performance of $\mathfrak{U}$adaBoost can be further improved with carefully selected Universum data.

3. The 𝔘adaBoost Algorithm

We first give an optimization objective for 𝔘adaBoost and show the proposed algorithm with the same procedure as AdaBoost. Then we compare 𝔘adaBoost with semi-supervised boosting algorithms, AdaBoost and 𝔘Boost.

3.1. Loss function

Boosting learns a strong classifier consisting of a voted combination of base learners, i.e. weak learners. The strong classifier is $sign[f(\mathbf{x})]$, where $f(\mathbf{x})$ is the linear combination of weak learners: $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$. For binary classification, $h_t(\mathbf{x}) \rightarrow \{+1, -1\}$ is weak classifier, $\alpha_t (\geq 0)$ is the weight for linear combination and T is the number of selected weak classifiers.

When Universum data is added into a supervised learning algorithm, it needs additional loss function or regularization term for Universum data. So does AdaBoost algorithm. We propose 𝔘adaBoost algorithm which contains Universum data to improve the performance of AdaBoost. A loss function $L(y, f(\mathbf{x}))$ is defined for the learning process of 𝔘adaBoost which contains two components corresponding to the labeled data L_1 and Universum data L_2 as:

$$L(y, f(\mathbf{x})) = \frac{1}{m}\sum_{i=1}^{m} L_1(y_i, f(\mathbf{x}_i)) + \frac{c}{2n}\sum_{j=1}^{n} L_2(f(\mathbf{x}_j^*)). \quad (4)$$

$c\ (\geq 0)$ is the trade-off parameter between the loss of labeled data and Universum data. It also can be regarded as the contribution of Universum data. Since we are trying to formulate our model as a boosting method, conventional boosting algorithms can be applied here, e.g. AdaBoost, GentleBoost [18], LogitBoost [18]. For simplicity, we use AdaBoost with the traditional exponential loss function for the labeled data while squared loss function for Universum data, i.e. $L_1(y, f(\mathbf{x})) = e^{-yf(\mathbf{x})}$, $L_2(f(\mathbf{x}^*)) = f(\mathbf{x}^*)^2$. Then Eq.(4) can be written as:

$$L(y, f(\mathbf{x})) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(\mathbf{x}_i)} + \frac{c}{2n}\sum_{j=1}^{n} f(\mathbf{x}_j^*)^2 \quad (5)$$

When c equals 0, the above equation becomes the standard AdaBoost.

Within the margin cost functional framework, $\rho_i = y_i f(\mathbf{x}_i)$ is defined as the margin of an labeled example $(\mathbf{x}_i, y_i)$ and then $L_1 = e^{-\rho}$ in AdaBoost. Considering the loss for Universum data, we define the margin for Universum sample $\mathbf{x}_j^*$ with $\rho_j^* = f(\mathbf{x}_j^*)$ and the corresponding loss $L_2 = \rho^{*2}$.

3.2. Learning

Since L is a convex function (both L_1 and L_2 are convex) and the available weak classifiers span a convex set, Eq.(5) is a convex optimization problem. Then the optimization problem will converge to a globally optimal solution. We adopt the functional gradient descent view of boosting to select the weak learner during each iteration of the boosting [18] [20]. According to the gradient descent principles, at the training step t, a weak classifier $h_t(\mathbf{x})$ will be selected to be added to the current ensemble classifier $f_{t-1}(\mathbf{x})$. This selected $h_t(\mathbf{x})$ should improve the objective value based on the loss function. Given the current learned strong classifier at iteration t, $f_t(\mathbf{x})$, the descent direction should be the gradient direction. Since the best weak classifier may not be available from the weak classifiers candidates (e.g. function family), the available best weak classifier can be obtained from the following equation. Note that we sometimes omit the argument $\mathbf{x}$ throughout this paper.

$$h^* = \arg\max_{h} - < \bigtriangledown L(y, f), h > \quad (6)$$

where $<, >$ is dot product, $< f, h >= \sum_i f(\mathbf{x}_i)h(\mathbf{x}_i)$. The gradient of convex function $L(y, f)$ based on Eq.(5) with respect to the current models $f(\mathbf{x})$ can be written as:

$$\bigtriangledown L(f)(\mathbf{x}) = \begin{cases} \frac{1}{m} y e^{-yf(\mathbf{x})}, & \mathbf{x} = \mathbf{x}_i \\ \frac{c}{n} f(\mathbf{x}^*), & \mathbf{x} = \mathbf{x}_j^*, \\ 0, & otherwise, \end{cases} \quad (7)$$

The sample weight is defined as

$$w_i = \frac{L'(\rho(f(\mathbf{x}_i), y_i))}{\sum_{\mathbf{x}_i} L'(\rho(f(\mathbf{x}_i), y_i))}, \text{ with } L_1' = -e^{-\rho}, L_2' = 2\rho^* \quad (8)$$

Then the best weak classifier h^* for the current learned strong classifier $f(x)$ should be

$$\begin{aligned} h^* &= \arg\max_{h} - < \bigtriangledown L(y, f), h > \\ &= -\frac{1}{m+n}\{\frac{1}{m}\sum_{i=1}^{m} -y_i h(\mathbf{x}_i) e^{-y_i f(\mathbf{x}_i)} + \frac{c}{n}\sum_{j=1}^{n} f(\mathbf{x}_j^*) h(\mathbf{x}_j^*)\} \\ &= -\frac{1}{m+n}\{\frac{1}{m}\sum_{i=1}^{m} -y_i w_i h(\mathbf{x}_i) + \frac{c}{n}\sum_{j=1}^{n} w_j^* h(\mathbf{x}_j^*)\} \end{aligned} \quad (9)$$

where $w_i = e^{-y_i f(\mathbf{x}_i)}$ and $w_j^* = f(\mathbf{x}_j^*)$ are the weights of labeled sample $\mathbf{x}_i$ and Universum data $\mathbf{x}_j^*$ based on Eq.(8).

Here we give a brief explanation of the weights for Universum samples. Since the optimal objective value for $\mathbf{x}_j^*$ is 0, 0 can be treated as its label. This is reasonable because it does not belong to any class of the problem while the optimal values for labeled samples should be their labels. Based on this explanation, Universum examples are misclassified during all the iterations since each weak classifier gives the prediction +1/-1. If its current predicted value $h_t(\mathbf{x}_j^*)$ (+1/-1) is the same as the previous iteration $h_{t-1}(\mathbf{x}_j^*)$, meaning that the previous error still exists, its weight increases. This is consistent with AdaBoost which amplifies the importance of those misclassified samples. On the other hand, if $h_t(\mathbf{x}_j^*) \neq h_{t-1}(\mathbf{x}_j^*)$, its weight decrease since $w_j^* = f(\mathbf{x}_j^*) = \sum_t \alpha_t h_t(\mathbf{x}_j^*)$. This implies that current weak classifier changes the prediction for this sample and decreases the ensemble predicate value (error).

The boosting algorithm should terminate as no h can be found to reduce $L(y,f)$, i.e. $-<\bigtriangledown L(y,f),h>\leq 0$. The optional step size, in the direction of h^*, is

$$\alpha^* = \arg\min_{\alpha} L(f(\mathbf{x}) + \alpha h^*(\mathbf{x})) \tag{10}$$

By the gradient descent method, the best α^* happens when the gradient $\bigtriangledown L(f+\alpha h^*) = 0$. That is

$$\begin{aligned}\frac{\partial L(f+\alpha h^*)}{\partial \alpha} &= \frac{1}{m}\sum_i -yh^* e^{-yf} e^{-y\alpha h^*} + \frac{c}{n}\sum_j (f+\alpha h^*)h^* \\ &= \frac{1}{m}\sum_i -yw_i h^* e^{-y\alpha h^*} + \frac{c}{n}\sum_j (w_j^* + \alpha h^*)h^* \\ &= \underbrace{\frac{1}{m}(\sum_{y_i \neq h^*(\mathbf{x}_i)} w_i e^{\alpha} - \sum_{y_i = h^*(\mathbf{x}_i)} w_i e^{-\alpha})}_{\text{AdaBoost for labeled data}} + \frac{c}{n}\sum_j (w_j^* + \alpha h^*)h^* \end{aligned} \tag{11}$$

When c equals 0, the optimization problem will become the standard AdaBoost. Then we can obtain the closed solution for α:

$$\alpha^* = \frac{1}{2}\ln\frac{1-\sum_{y_i\neq h^*(\mathbf{x}_i)} w_i}{\sum_{y_i\neq h^*(\mathbf{x}_i)} w_i}. \tag{12}$$

Algorithm 1 𝔘adaBoost: AdaBoost with the Universum

```
1: Input:
     Training samples: $\mathcal{X}_{\mathcal{L}}$ and $\mathcal{X}_{\mathfrak{U}}$;
     The number of weak classifiers: $T$;
     Realization parameter: $c$.
2: Initialization:
     set $t = 0$, $f_t(\mathbf{x}) = 0$;
     $\forall \mathbf{x} \in \mathcal{X}_{\mathcal{L}} : w_{\mathcal{L}}(\mathbf{x}) = \frac{1}{m}$ and $\forall \mathbf{x} \in \mathcal{X}_{\mathfrak{U}} : w_{\mathfrak{U}}(\mathbf{x}) = \frac{1}{n}$
3: for $t = 1$ to $T$ do
4:     Normalize the weights
       $\forall \mathbf{x} \in \mathcal{X}_{\mathcal{L}} : w_{\mathcal{L}}(\mathbf{x}) \leftarrow w_{\mathcal{L}}(\mathbf{x}) / \sum_{\mathbf{x}\in\mathcal{X}_{\mathcal{L}}} w_{\mathcal{L}}(\mathbf{x})$
       $\forall \mathbf{x} \in \mathcal{X}_{\mathfrak{U}} : w_{\mathfrak{U}}(\mathbf{x}) \leftarrow w_{\mathfrak{U}}(\mathbf{x}) / \sum_{\mathbf{x}\in\mathcal{X}_{\mathfrak{U}}} w_{\mathfrak{U}}(\mathbf{x})$
5:     Find the best weak classifier by Eq.(9)
6:     Find the optimal step $\alpha^*$ by Eq.(11) (13) (14)
7:     Update the weights
       $\forall \mathbf{x} \in \mathcal{X}_{\mathcal{L}} : w_{\mathcal{L}}(\mathbf{x}) \leftarrow w_{\mathcal{L}}(\mathbf{x}) e^{-yh^*(\mathbf{x})}$
       $\forall \mathbf{x} \in \mathcal{X}_{\mathfrak{U}} : w_{\mathfrak{U}}(\mathbf{x}) \leftarrow w_{\mathfrak{U}}(\mathbf{x}) + h^*(\mathbf{x})$
8:     Update the strong classifier: $f_{t+1} = f_t + \alpha^* h^*$
9: end for
10: Output: The final strong classifier $sign[f_T(\mathbf{x})]$.
```

However, there is no closed solution in Eq.(11), so line search is performed to find the optimal step size. In practice Newton-Raphson method is used for its fast speed. The iteration at $n+1$-th is

$$\alpha_{n+1} = \alpha_n - \frac{\bigtriangledown L(\alpha_n)}{\bigtriangledown^2 L(\alpha_n)} \tag{13}$$

The second order gradient is

$$\bigtriangledown^2 L(f+\alpha h^*) = \frac{1}{m}\sum_{i=1}^{m} w_i e^{-y\alpha h^*} + c \tag{14}$$

The updated strong classifier is $f_{t+1} = f_t + \alpha^* h^*$.

During the iterations, some variables can be pre-saved to speed up the calculation of the first derivative and the second derivative. Experiments show that the iteration stops after only several steps. The framework of 𝔘adaBoost is presented in Algorithm 1.

3.3. Comparison with Semi-supervised Boosting

𝔘adaBoost is different from semi-supervised boosting algorithms because Universum data does not belong to any class of the problem while unlabeled data can be treated as either positive samples or negative samples. That is to say, there is an explicit label for unlabeled sample even we cannot know during the learning procedure. The label of Universum sample is explicitly known which can be treated as 0 during the training.

Besides that, the difference of conventional semi-supervised boosting algorithms and 𝔘adaBoost is clear from the loss function. Specifically, a pseudoclass [21] or a pseudomargin [22] is always introduced to unlabeled data in semi-supervised algorithms. The pseudoclass label of an unlabeled point $\mathbf{x}$ is typically defined as $\widetilde{y} = sign[f(\mathbf{x})]$ and its corresponding pseudomargin is $\widetilde{y}sign[f(\mathbf{x})] = |f(\mathbf{x})|$ [22] [21]. With pseudoclass or pseudomargin, semi-supervised boosting algorithms can be treated as a supervised learning problem. In comparison, 𝔘adaBoost omits the label for Universum data.

3.4. Training Efficiency Evaluation on AdaBoost, 𝔘Boost and 𝔘adaBoost

Training efficiency (e.g. training time) is an important issue to be considered in Universum based boosting. In order to select effective Universum data and proper parameter for the loss of Universum data, a large number of experiments are needed (e.g. cross validation). Fast training procedure is beneficial to better experimental design. Training speed is also of great concern when a large amount of weak classifier candidates and training iterations are evaluated, e.g. Haar-like feature in face detection application [23]. Here we compare the training speed of AdaBoost, 𝔘Boost and 𝔘adaBoost. Boosting based algorithms are trained through a number of iterations. Thus training time is determined by the cost of one iteration and the number of iterations.

AdaBoost is the fastest in one iteration training for weak classifier selection since only labeled samples are included. For the Universum based boosting algorithms, 𝔘adaBoost and 𝔘Boost share the same weak classifier candidates generation step. 𝔘adaBoost only needs to update one weak classifier coefficient α in each training iteration while all coefficients for the pre-selected weak classifiers in 𝔘Boost are updated. Thus 𝔘adaBoost is faster than 𝔘Boost in each iteration.

In terms of number of iterations, both 𝔘Boost and 𝔘adaBoost have a faster convergence speed than AdaBoost. 𝔘Boost needs the least number of iterations to converge because column generation is used. Fig. 2 shows that three algorithms converge at about 100 iterations. Experiment is performed on MNIST data set (see description in section 5). Although $\mathfrak{U}$Boost obtains the best training performance, the gaps with $\mathfrak{U}$adaBoost is marginal. Overall, to reach the convergence point, the total time cost by training procedures for $\mathfrak{U}$Boost is much more than $\mathfrak{U}$adaBoost. Experiments show that the speed of $\mathfrak{U}$adaBoost is more than 10 times faster than $\mathfrak{U}$Boost. Specific total time for training a classifier is shown in Table 1.

4. Practical Conditions for $\mathfrak{U}$adaBoost

Although Universum data has been applied into a lot of applications and have been analyzed, it needs carefully selected since not all the generated Universum samples are effective and informative. Cherkassky *et al.* [19] took histogram as an analysis tool and gave practical conditions to verify the effectiveness of Universum data under SVM classifier. First, an optimal SVM classifier will be trained with the labeled data. Then they project the training data and Universum data onto the normal direction vector of the trained hyperplane. Finally, the relationships of training data and Universum data are analyzed. The histogram of projections of effective Universum data should be symmetric relative to the SVM decision boundary and have a wide distribution between margin borders (+1/-1).

These conditions are designed for SVM based learning. When they are applied into boosting based learning, two differences happen. The first one is the standard boosting does not need parameters to tune compared to the standard SVM which has one parameter $C_{\mathfrak{L}}$ in Eq.(1). When Universum data is added, least square version of $\mathfrak{U}$-SVM and $\mathfrak{U}$adaBoost both need another parameter of the trade-off losses between labeled data and Universum data. The original $\mathfrak{U}$-SVM still needs another parameter ε in Eq.(1). That is to say, $\mathfrak{U}$adaBoost is easier to use than $\mathfrak{U}$-SVM. The second one is that AdaBoost is not an explicit margin controlled algorithm. It only considers the training loss while SVM gives an explicit margin in the optimization problem in Eq.(1). AdaBoost does not maximize the minimum margin but has some relations about the average and division of the margin [24]. In comparison, SVM separates the two classes with a clear margin [-1 +1].

Based on above analysis, the same Universum data matching above conditions will not be appropriate for $\mathfrak{U}$adaBoost. But we can still analyze Universum data with the tool of histogram. As in [19], the distribution of effective Universum data should be symmetric relative to decision hyperplane $\mathbf{w}^\top\mathbf{x} + b = 0$. In boosting, the decision function is $f(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x}) + b$, with $b = 0$ from section 3.2. In practice, we can project the available Universum data onto the pre-trained AdaBoost classifier and choose those samples meeting the conditions above. For a separable problem, AdaBoost can also give a clear margin (similar to Fig. 5). The following experiments will show that the performance of $\mathfrak{U}$adaBoost can be further improved by selecting effective Universum data.

5. Experiment

5.1. Experimental setup

Handwritten digits sets are always used for evaluating the algorithms with Universum data [15] [16] [3]. The reason is that if we take two digits for classification, the other digits data can be naturally used as Universum data. We experiment $\mathfrak{U}$adaBoost on two handwritten sets: MNIST and USPS, and a computer vision application: Gender Recognition. The original feature provided in the data sets (gray value) is used. Decision stump is applied as weak classifiers in all tests and the maximum number of iterations T_{max} is limited to 1000. Note that most of the boosting algorithms converge at less than 100 iterations from the experiments. There is only one parameter c for the loss ratio between labeled data and Universum data. c is validated from $\{2^{-17}, 2^{-15}, 2^{-13}, 2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}\}$. The best performance with the highest accuracy on validation set will be selected. The experiments are run on all the data sets for 10 times, and average test errors with standard deviations are reported.

5.2. Performance Comparison

MNIST MNIST handwritten digits data set contains 10 categories of digits with 1,000 for each digit and 10,000 in total. It has split the total data into training set and test set. Each digit is represented by a 28 × 28 image. Sample images are shown in Fig. 3.

Since both $\mathfrak{U}$SVM and $\mathfrak{U}$Boost showed their performance of classifying digits '5' and '8' with digits '3' and '6' for Universum data, we also take this option. The training and validation samples are randomly selected from the original training set. The original test set is used for testing. We use all the training examples of digits '3' and '6' for Universum data (12,094 examples). Four experiments are performed with different sizes of training and validation samples. The size of validation set is the same as training set which is selected from {200, 400, 600, 800}. The classification accuracies and standard deviations are reported in Table 2. In addition, we fix the size of validation set with 400 and change the size for training set. The results are shown in Table 3. Both Table 2 and 3 show that $\mathfrak{U}$adaBoost outperformed AdaBoost under different situations. Compared with $\mathfrak{U}$Boost, our method achieves better performances for the most of cases. Still we notice that the performance of $\mathfrak{U}$Boost is unstable. Sometimes, it is even inferior to AdaBoost.

USPS The USPS data set contains 16×16 images for 10 categories of digits. Sample images are shown in Fig. 3. Like the experiment of MNIST data set, we classify digits '5' vs. '8' with '3' and '6' as Universum data. The original test is used for testing. Four experiments are taken with different sizes of training examples which are selected from {100, 200, 400, 600}. The validation set contains 200 examples. All the training examples of digits '3' and '6' are used as Universum data, containing 1,287 examples. The classification results are shown Table 4. Our proposed algorithm achieves better performances over AdaBoost and 𝔘Boost in most cases.

Another experiment shows the effectiveness of different sizes of Universum data. We use 400 examples for training and 400 examples for validation. The Universum data set is selected with different sizes (100, 300, 500, 800). From Table 5, we can see that, the performances of 𝔘adaBoost are further improved with more Universum data.

Gender Recognition We test 𝔘adaBoost on a gender recognition task [14]. The data set contains 113 males and 20 females (20 images for per person). Sample images are shown in Fig. 4. The colored face images are converted to 8-bit gray level images and scaled to 45×50 sizes without extra preprocessing. The feature length for each training example is 2,250.

Following the experimental setting of [14], a set of 52 individuals (32 males and 20 females) is randomly selected, containing 13 individuals for training (8 males and 5 females) and the remaining 39 individuals for test. The training and validation sets are randomly selected from 52 individuals (one image for one individual) consisting of 13 training samples, 13 validation samples and 39 test samples. Universum data is generated by random average [14] shown in Fig. 4 (the average value of one male sample and one female sample). Experimental results with different sizes of Universum data are shown in Table 6. Similar to previous results, the performances of the standard AdaBoost are improved by Universum data.

5.3. Universum Data Selection

In order to further take advantage of Universum data, we refine the Universum data by ignoring the Universum samples whose final confidence scores are out of the margin of the pre-trained classifier. For example, for the setup of Table 3, the histograms of projections over training data and Universum data are shown in Fig. 5. The margin is about [-0.18 0.18] which is different from SVM based margin. The majority of projections of Universum samples are located into the margin borders. We refine the original Universum data and reject about 3,000 Universum samples. Then we retrain the classifier with this newly generated Universum data. The results are shown in the last row of Table 3. It shows that the performance of 𝔘adaBoost can be further improved. The similar results can be seen in Table 4.

In order to investigate the effectiveness of different kinds of Universum data, additional experiments are conducted on three kinds of Universum data. For handwritten digits classification in MNIST data, digits '1', '3' and '6' are chosen as Universum data to classify digits '5' and '8'. The training/validation set size is 200 (100 per class); Universum set size is 1000 and test set size is 1866. Experimental results in Table 7 show that 𝔘adaBoost outperforms AdaBoost with all the three Universum data, and digits '3', '6' make it better. The observation is the same as 𝔘-SVM in [19]. Fig. 6 shows the histograms of projection for the three kinds of Universum data. This confirms the result of Table 7 that digit '1' is not as good as digits '3' and '6' for Universum data because its histogram of projections is more biased and a large amount of them are out of the decision boundary.

5.4. Remarks and Conclusion

𝔘adaBoost follows the same procedures as AdaBoost while the model of 𝔘Boost is based on AdaBoost-CG. The classification performance of AdaBoost-CG is similar to the standard stage-wise AdaBoost. There is no theoretical evidence to clarify which one is better [17]. Above experiments show such conclusion still remains when they meet Universum data. During the experiments, we find that 𝔘Boost is sensitive to parameters. Inappropriate parameters (i.e. c and D in Eq.(3)) will result in large decreases of performance. Only one parameter in 𝔘adaBoost (i.e. c in Eq.(5)) makes it more practical to use. Finally, with carefully selected Universum data, it is easy to predict the improved performance of AdaBoost with the help of Universum data.

6. Conclusion

This paper presented a new algorithm 𝔘adaBoost which improved the performance of AdaBoost by taking advantage of Universum data. Explicit weighting schemes were derived by functional gradient descent which made 𝔘adaBoost similar speed to AdaBoost. By analyzing the distribution of Universum data on AdaBoost classifier, more informative and effective Universum data can be obtained. Experiments showed that the performance of 𝔘adaBoost can be improved further with the refined Universum data.

References

1. Vapnik, V.. Estimation of dependences based on empirical data. Springer-Verlag New York Inc; 2006.
2. Weston, J., Collobert, R., Sinz, F., Bottou, L., Vapnik, V.. Inference with the universum. In: *Proceedings of International Conference on Machine learning (ICML2006)*. 2006:1009–1016.
3. Sinz, F., Chapelle, O., Agarwal, A., Schölkopf, B.. An analysis of inference with the universum. In: *Advances in Neural Information Processing Systems (NIPS2008)*. 2008:1369–1376.
4. Zhang, D., Wang, J., Wang, F., Zhang, C.. Semi-supervised classification with universum. In: *Proceedings of SIAM International Conference on Data Mining (SDM2008)*. 2008:323–333.
5. Chen, X., Chen, S., Xue, H.. Universum linear discriminant analysis. *Electronics letters* 2012;48(22):1407–1409.
6. Qi, Z., Tian, Y., Shi, Y.. Twin support vector machine with universum data. *Neural Networks* 2012;36:112–119.
7. Dhar, S., Cherkassky, V.. Cost-sensitive universum-svm. In: *Proceedings of International Conference on Machine Learning and Applications (ICMLA2012)*; vol. 1. 2012:220–225.
8. Qi, Z., Tian, Y., Shi, Y.. Regularized multiple-criteria linear programming with universum and its application. *Neural Computing and Applications* 2014;24(3-4):621–628.
9. Duan, L., Xu, D., Tsang, I.W.. Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Trans Neural Networks and Learning Systems* 2012;23(3):504–518.

10. Hao, X., Zhang, D.. Ensemble universum svm learning for multimodal classification of alzheimers disease. In: *Mach. Learn. in Med. Imaging*; vol. 8184. 2013:227–234.
11. Zhang, D., Wang, J., Si, L.. Document clustering with universum. In: *Proceedings of International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR2011)*. 2011:873–882.
12. Peng, B., Qian, G., Ma, Y.. Recognizing body poses using multilinear analysis and semi-supervised learning. *Pattern Recogn Letters* 2009;30(14):1289–1294.
13. Peng, B., Qian, G., Ma, Y.. View-invariant pose recognition using multilinear analysis and the universum. In: *Proceedings of Advances in Visual Computing*; vol. 5359. 2008:581–591.
14. Bai, X., Cherkassky, V.. Gender classification of human faces using inference through contradictions. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN2008)*. 2008:746 –750.
15. Chen, S., Zhang, C.. Selecting informative universum sample for semi-supervised learning. In: *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI2009)*; vol. 6. 2009:1016–1021.
16. Shen, C., Wang, P., Shen, F., Wang, H.. Uboost: Boosting with the universum. *IEEE Trans Pattern Anal Mach Intell* 2012;34(4):825–832.
17. Shen, C., Li, H.. On the dual formulation of boosting algorithms. *IEEE Trans Pattern Anal Mach Intell* 2010;32(12):2216–2231.
18. Friedman, J., Hastie, T., Tibshirani, R.. Additive logistic regression: a statistical view of boosting. *Ann Statist* 2000;28(2):337–407.
19. Cherkassky, V., Dhar, S., Dai, W.. Practical conditions for effectiveness of the universum learning. *IEEE Trans Neural Networks* 2011;22(8):1241–1255.
20. Mason, L., Baxter, J., Bartlett, P., Frean, M.. Boosting algorithms as gradient descent in function space. In: *Advances in Neural Information Processing Systems (NIPS1999)*; vol. 12. 1999:512–518.
21. d'Alché Buc, F., Grandvalet, Y., Ambroise, C.. Semi-supervised marginboost. In: *Advances in Neural Information Processing Systems (NIPS2002)*; vol. 14. 2002:553–560.
22. Bennett, K., Demiriz, A., Maclin, R.. Exploiting unlabeled data in ensemble methods. In: *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2002)*. 2002:289–296.
23. Viola, P., Jones, M.J.. Robust real-time face detection. *Int J Comp Vis* 2004;57(2):137–154.
24. Rätsch, G., Warmuth, M.. Efficient margin maximizing with boosting. *J Mach Learn Res* 2005;6:2131–2152.

List of Figures

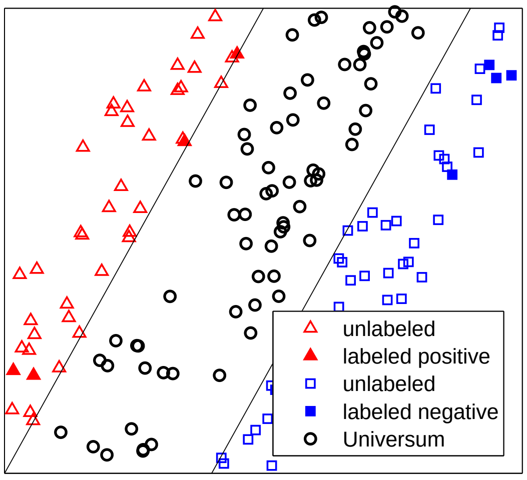


Figure 1: The comparison of semi-supervised learning problem and Universum based learning problem. The former takes labeled positive, labeled negative and unlabeled samples for learning, and the latter learns models from labeled positive, labeled negative and Universum samples.

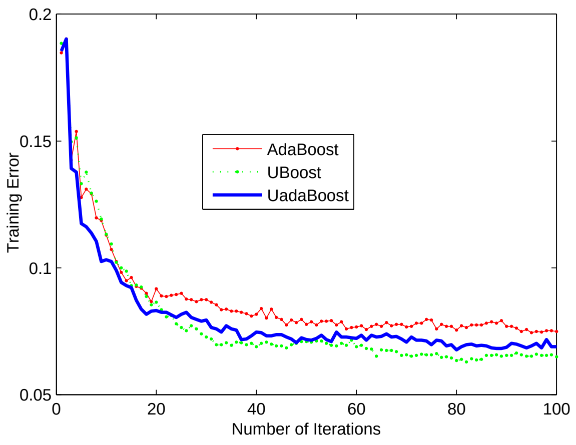


Figure 2: Training error of AdaBoost, 𝔘Boost and 𝔘adaBoost on MNIST data set. The training set size is 200 and the validation set size is 400.

(a) (b)

Figure 3: Examples of handwritten digits from two data sets. (a) MNIST (b) USPS.

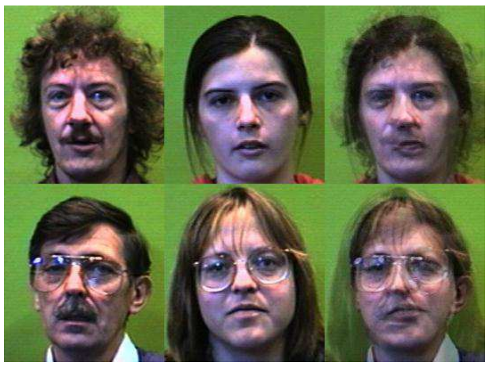


Figure 4: Examples of male and female faces (column 1 and 2) and the corresponding Universum samples (column 3) obtained by averaging the two images from column 1 and 2 respectively.

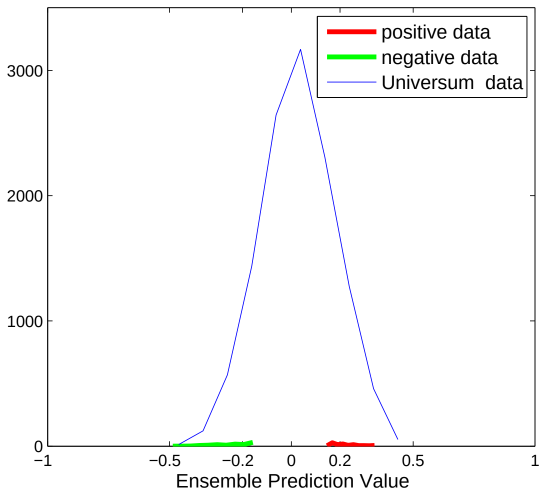
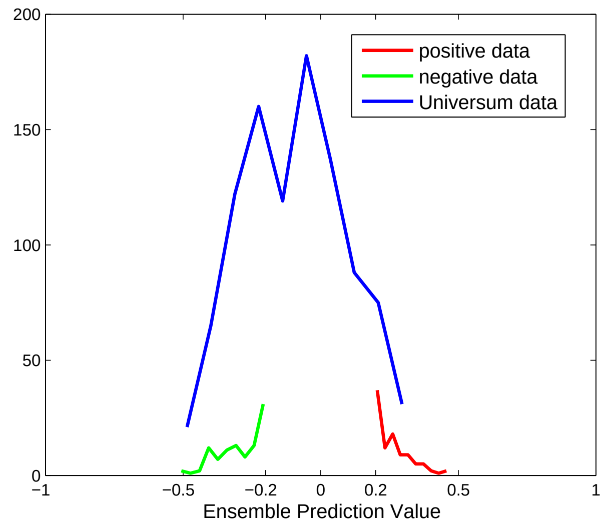
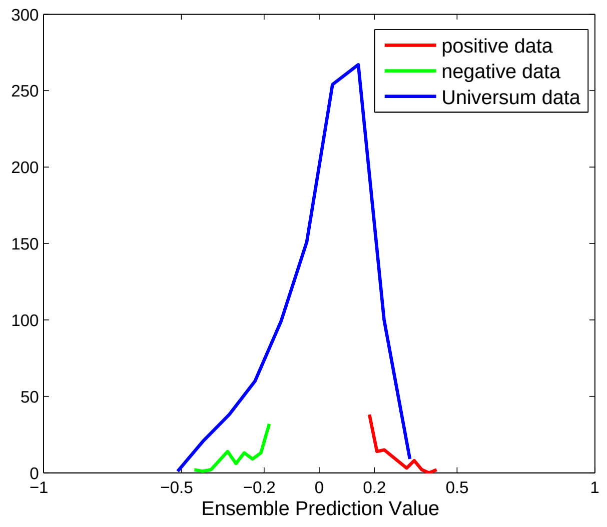


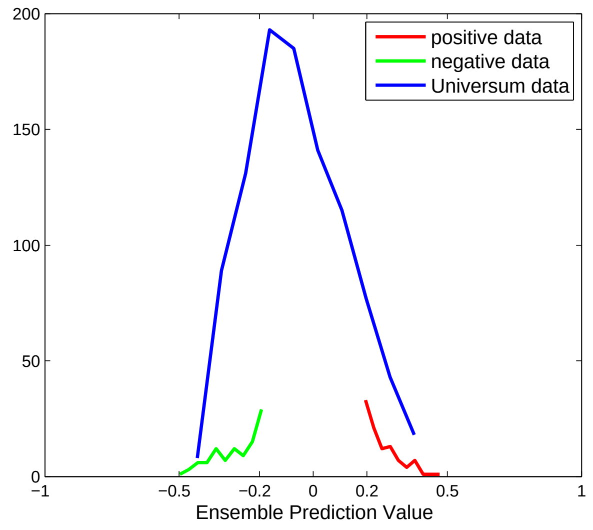
Figure 5: Histogram of ensemble prediction over labeled training data and Universum data. The vertical axis is the number of training samples. The model is trained by 𝔘adaBoost on MNIST data set.



(a)



(b)



(c)

Figure 6: Histogram of ensemble prediction over labeled training data and Universum data. The vertical axis is the number of training samples. The model is trained by 𝔘adaBoost on MNIST data set. The Universum data comes from digits '1' (a), '3' (b) and '6' (c) respectively.

List of Tables

Table 1: Training time (s) comparisons between $\mathfrak{U}$Boost and $\mathfrak{U}$adaBoost. Experiments are performed on MMIST and USPS data set respectively. In MNIST data set (MNIST 1 and MNIST 2), the train set size is 100 and 400 respectively, and the validation set size is 400. In USPS data set (USPS 1 and USPS 2), one is with 400 training samples, 400 validation samples and the other is with 600 training samples, 200 validation samples.

Method	MNIST 1	MNIST 2	USPS 1	USPS 2
$\mathfrak{U}$Boost	2.05×10^4	2.96×10^4	1.04×10^4	1.55×10^4
$\mathfrak{U}$adaBoost	2.00×10^3	2.08×10^3	3.45×10^2	4.10×10^2

Table 2: Experimental results (test error rates (%) mean ± standard deviation) on MMIST data set with different training data sizes. The validation set size is the same as training set size. Digits '5' and '8' are used as labeled data, and digits '3' and '6' as Universum data (12,094 samples).

Method	200	400	600	800
AdaBoost	7.22 ± 0.69	5.38 ± 0.58	5.86 ± 0.56	4.97 ± 0.30
$\mathfrak{U}$Boost	7.42 ± 1.14	5.52 ± 0.88	5.25 ± 0.45	4.66 ± 0.40
$\mathfrak{U}$adaBoost	7.19 ± 0.76	5.36 ± 0.46	5.78 ± 0.56	4.79 ± 0.40

Table 3: Experimental results (test error rates (%) mean ± standard deviation) on MMIST data set with different training data sizes. The validation set size is 400. Digits '5' and '8' are used as labeled data, and digits '3' and '6' as Universum data (12,094 samples). The last line shows the results of 𝔘adaBoost with the refined Universum data (9657 samples).

Method	100	200	300	400
AdaBoost	8.33 ± 1.15	7.49 ± 0.62	6.63 ± 0.76	5.38 ± 0.58
𝔘Boost	8.83 ± 1.15	7.50 ± 0.59	6.40 ± 0.69	5.52 ± 0.88
𝔘adaBoost	8.17 ± 1.06	7.33 ± 0.51	6.42 ± 0.46	5.36 ± 0.46
𝔘adaBoost*	8.10 ± 1.18	7.29 ± 0.59	6.33 ± 0.49	5.28 ± 0.30

Table 4: Experimental results (test error rates (%) mean ± standard deviation) on USPS data set with different training data sizes. The validation set size is 200. Digits '5' and '8' are used as labeled data, and digits '3' and '6' as Universum data. The last line shows the results of 𝔘adaBoost with the refined Universum data (950 samples).

Method	100	200	400	600
AdaBoost	4.84 ± 1.35	4.06 ± 0.71	3.37 ± 0.41	3.05 ± 0.51
𝔘Boost	4.44 ± 0.86	4.02 ± 0.92	3.77 ± 0.80	3.51 ± 0.80
𝔘adaBoost	4.66 ± 1.26	4.03 ± 0.50	3.28 ± 0.36	3.02 ± 0.41
𝔘adaBoost*	4.66 ± 1.41	4.00 ± 0.64	3.20 ± 0.33	3.01 ± 0.43

Table 5: Experimental results (test error rates (%) mean ± standard deviation) on USPS data set with different Universum data sizes. Digits '5' and '8' are used as labeled data, and digits '3' and '6' as Universum data. The training set size and the validation set size are both 400 (200 per class).

Method	100	300	500	800
AdaBoost	3.17 ± 0.45	3.17 ± 0.45	3.17 ± 0.45)	3.17 ± 0.45)
𝔘Boost	3.74 ± 0.35	3.60 ± 0.60	3.74 ± 0.45)	3.48 ± 0.49)
𝔘adaBoost	3.11 ± 0.46	3.05 ± 0.45	3.05 ± 0.41)	3.02 ± 0.54)

Table 6: Experimental results (test error rates (%) mean ± standard deviation) of Gender Recognition with different sizes of Universum data. Universum data is generated by random average. The training set size and the validation set size are both 13 (one image for one individual).

Method	100	500	1000
AdaBoost	23.85 ± 14.51	23.85 ± 14.51	23.85 ± 14.51
𝔘Boost	23.62 ± 7.76	21.54 ± 9.76	23.07 ± 8.55
𝔘adaBoost	20.00 ± 10.79	20.76 ± 6.67	23.07 ± 9.75

Table 7: Experimental results (test error rates (%) mean ± standard deviation) on MNIST data set with different kinds of Universum data (digits '1', '3' and '6').

Method	digit '1'	digit '3'	digit '6'
AdaBoost	6.82 ± 0.71	6.82 ± 0.71	6.82 ± 0.71
𝔘adaBoost	6.79 ± 0.91	6.66 ± 0.68	6.76 ± 0.69